Supporting the Automatic Construction of Entity Aware Search Engines

Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, Paolo Papotti
Dipartimento di Informatica e Automazione
Università degli Studi Roma Tre - Italy
[blanco,crescenz,merialdo,papotti]@dia.uniroma3.it

ABSTRACT

Several web sites deliver a large number of pages, each publishing data about one instance of some real world entity, such as an athlete, a stock quote, a book. Although it is easy for a human reader to recognize these instances, current search engines are unaware of them. Technologies for the Semantic Web aim at achieving this goal; however, so far they have been of little help in this respect, as semantic publishing is very limited.

We have developed a method to automatically search on the web for pages that publish data representing an instance of a certain conceptual entity. Our method takes as input a small set of sample pages: it automatically infers a description of the underlying conceptual entity and then searches the web for other pages containing data representing the same entity. We have implemented our method in a system prototype, which has been used to conduct several experiments that have produced interesting results.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Algorithms

1. INTRODUCTION

There is an increasing number of web sites that deliver "data rich" pages, where the published information is organized according to an implicit schema. These pages usually contain high quality data that represent instances of some *conceptual entity*. Consider web sites that publish information about popular sport events, or web sites that publish financial information: their pages embed data that describe instances of conceptual entities such as athlete, match, team, or stock quote, company, and so on. To give a concrete example, observe the web pages in Figure 1. Each of them contains data describing one instance of the BASKETBALLPLAYER conceptual entity.

For the sake of scalability of the publishing process, the structure of pages and navigation paths of these web sites are fairly regular. Within each site, pages containing the same *intensional* information, i.e. instances of the same conceptual entity, offer the same type of information, which is organized according to a common template. In addition, the access paths (e.g. from the home page) to these pages obey to a common pattern. Again from our basketball example: in a given web site, the pages of two distinct players contains data—such as name, date of birth, and so on—that are organized according to the same page template. Also, these pages can be reached following similar navigation paths from the home page.

Although it is easy for a human reader to recognize these instances, as well as the access paths to the corresponding pages, current search engines are unaware of them. Technologies for the Semantic web aim at overcoming these limitations; however, so far they have been of little help in this respect, as semantic publishing is very limited.

To overcome this issue, search engine companies are providing facilities to build personal search engines that can be specialized over specific domains. A prominent example is Google Co-op, a Google facility that allows users to indicate sets of pages to be included in the personal search engine, and to assign a label (*facet* in the Google terminology) to them. Labels aim at providing a semantic meaning to the page contents, and are used to enhance the search engine querying system. For data rich pages, labels typically represent a name for underlying conceptual entity. For example, a user interested in building a personal search engine about the basketball world can provide the system with web pages containing data about players, such as those in Figure 1, and then she can associate them with the label BASKETBALLPLAYER to indicate that they contain data about instances of the basketball player conceptual entity. An alternative approach with similar goals is based on mass labeling facilities, such as `del.icio.us` or `reddit.com`, which allow users to collaboratively annotate pages with labels.

We observe that although these approaches support users in the definition of search engines that are somehow aware about the presence of instances of a given entity, the issue of gathering the relevant pages must be performed manually by the user.

This paper proposes an original and effective domain independent solution to tackle the issue of the *page gathering task*. We believe that our method can help the above facilities scaling, as it automatically discovers pages containing data that represent instances of a given conceptual entity.

Our method takes as input a small set of sample pages from distinct web sites: it only requires that the sample pages contain data about an instance of the conceptual entity of interest. Then, leveraging redundancies and structural regularities that locally occur on the web, our method automatically discovers pages containing data about other instances of the conceptual entity exemplified by the

WIDM'08, October 30, 2008, Napa Valley, California, USA.

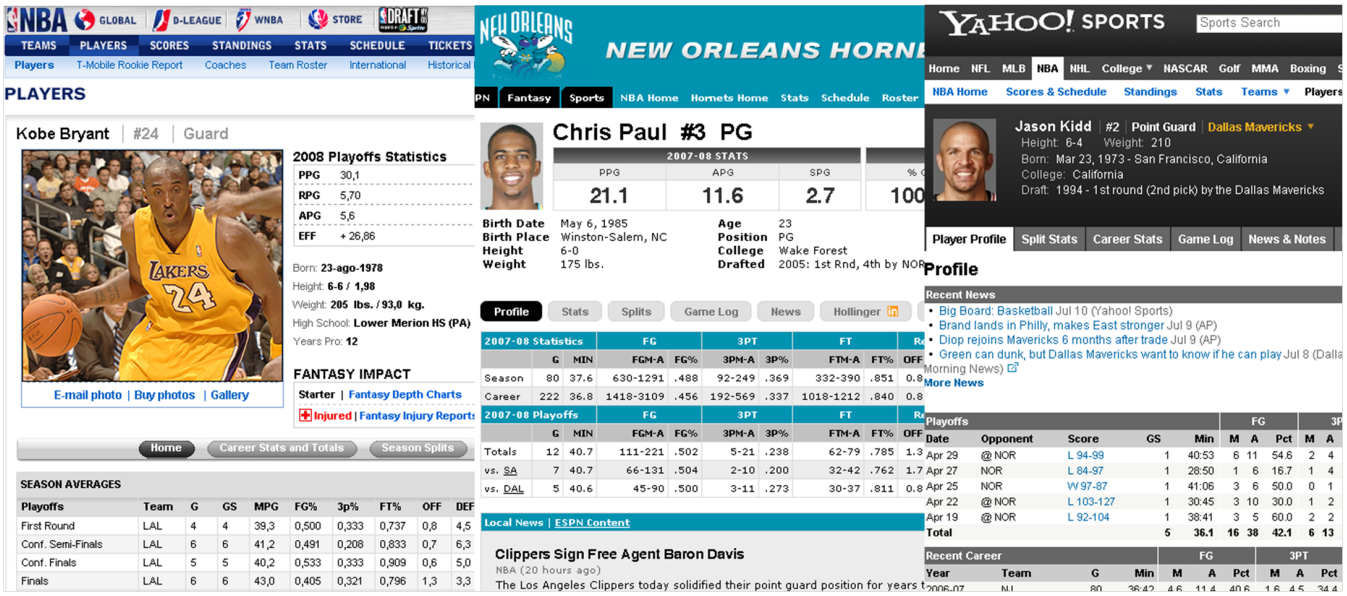

Figure 1: Web pages representing instances of the BASKETBALLPLAYER **conceptual entity.**

input samples, as follows.

- it crawls the web sites of the input sample pages to collect pages with data about other instances of the conceptual entity of interest;
- from these pages, it automatically extracts a description of the entity exemplified by the sample pages;
- using the information computed in the previous steps, it launches web searches to discover new pages. The results of these searches are analyzed using the entity description. Pages representing valid instances of the target entity are stored, and are used to recursively trigger the process.

It is important to notice that our technique has a different semantics with respect to the "similar pages" facility offered by search engines. Given as input two web pages from two different web sites describing the basketball players "Kobe Bryant" and "Bill Bradley", our method aims at retrieving many web pages that are similar at the intensional level, e.g. pages about other basketball players, not necessarily the same two sample players.
The rest of the paper is organized as follows. Section 2 provides a brief overview of our method that, after a discussion on related work in Section 3, is detailed in Sections 4 and 5. Section 6 illustrates the result of the experiments we have conducted to evaluate the effectiveness of the approach. Section 7 presents our concluding remarks and future work.

2. OVERVIEW

The ultimate goal of our method is to automatically discover web pages that contain data describing instances of a given conceptual entity. We assume that the user provides as input a few input sample pages. It is not important that the sample pages contain data about the same instance; we only require they come from different web sites, and they contain data that represent instances of the same conceptual entity. Pages such as those in Figure 1 could be used as input to collect pages with data about instances of the BASKETBALLPLAYER conceptual entity.

Searching Entity Pages within One Site. The first step of our method is to search the target pages within the web sites of each sample page. This task is performed by INDESIT, a crawling algorithm designed to drive a scan of a given web site toward pages sharing the same structure of an input seed page [5].

INDESIT relies on the observation that, within a large web site, pages offering a description of the same conceptual entity (e.g., BASKETBALLPLAYER) usually share a common template and similar access paths.

INDESIT efficiently navigates the web site to collect pages containing lists of links toward pages which are structurally similar to the seed page. Following these links it gathers pages with the type of information of the seed. With respect to our running example, the output of INDESIT is the set of basketball player pages published in the web sites of each sample page.

Learning a Description of the Conceptual Entity. As a second step, our method computes a description for the target conceptual entity. To this end, we rely on the observation that pages containing data about instances of the same conceptual entity share a common set of characterizing keywords that appear in the page template.

In our approach, the description of a conceptual entity is then composed by a set of *keywords* that are extracted from the set of terms that lay on the templates of the input sample pages. Our experiments show that these keywords effectively characterize the overall conceptual domain of the entity with very promising results.

Given a set of structurally similar pages returned by INDESIT, the entity description is generated by computing the terms that belong to the corresponding template. This task is performed by analyzing the set of terms that occur in the pages and by removing those elements that belong also to the "site template", i.e. to that portion of the template that is shared by every pages in the site. In this way, from each sample page a set of terms is extracted. Terms that are shared in the templates of different web sites are then selected as keywords for the entity description.

Triggering new Searches on the Web. The results produced by the initial INDESIT executions and the keywords in the entity description are used to propagate the search on the web. This step is done by the OUTDESIT algorithm, which issues a set of queries against a search engine and elaborates the results in order to select only those pages that can be considered as instances of the target entity. Then, the selected pages are used as seeds to trigger again an INDESIT scan, and the whole process is repeated until new pages are found.

To correctly expand the search on the web, we need to address two main issues. First, we have to feed the search engine with keywords that are likely to produce new pages representing instances of the input entity. Second, as these pages will be used to run a new instance of INDESIT, we have to filter them in order to choose those that really correspond to instances of the conceptual entity of interest.

To generate the keywords to be submitted to the search engine we adopt a simple yet effective solution. As we are searching for instances of a given entity, we need values that work as identifiers for the instances of the entity. We observe that, since pages are designed for human consumption, the anchors associated with the links to our instance pages usually satisfy these properties: they are expressive, and they univocally identify the instance described in the target page. In our example, the anchor to a player page usually corresponds to the name of the athlete. Therefore, we issue a number of queries against a search engine, where each query is composed by the anchor of a link to one of the pages retrieved by the previous INDESIT execution. Also, to focus the search engine toward the right domain, each query is completed with keywords from the entity description.

As search results typically include pages that are not suitable for our purposes, we filter the off-topic pages by requiring that the keywords of the entity description are contained in their templates.

The three steps described above are repeated to collect new relevant pages: the results that are selected from each search are used as INDESIT seeds to gather further pages and to trigger new searches.

3. RELATED WORK

Our method is inspired to the pioneering DIPRE technique developed by Brin [7]. With respect to DIPRE, which infers patterns that occur locally within single web pages to encode tuples, we infer global access patterns offered by large web sites containing pages of interest. DIPRE also inspired several web information extraction techniques [1, 3]. Compared to our approach these appraches are not able to exploit the information offered by data rich pages. In fact, they concentrate on the extraction of facts: large collections of named-entities (such as, for example, names of scientists, politicians, cities), or simple binary predicates, e.g. *born-in(politician, city)*. Moreover, they are effective with facts that appear in well-phrased sentences, whereas they fail to elaborate data that are implied by web page layout or mark-up practices, such as those typically published in web sites containing data rich pages.

Our work is also related to researches on focused crawlers (or topical crawlers) [9, 20, 19], which face the issue of efficiently fetching web pages that are relevant to a specific topic. Focused crawlers typically rely on text classifiers to determine the relevance of the visited pages to the target topic. Page relevance and contextual information—such as, the contents around the link, the lexical content of ancestor pages—are used to estimate the benefit of following URLs contained in the most of relevant pages. Although focused crawlers present some analogy with our work, our goal is different as we aim at retrieving pages that publish the same type of information, namely, pages containing data that represent instances of the conceptual entity exemplified by means of an input set of sample pages.

Vidal et al. present a system, called GOGETIT! that takes as input a sample page and an entry point to a web site and generates a sequence of *URL patterns* for the links a crawler has to follow to reach pages that are structurally similar to the input sample [22], therefore their approach is limited to address the issue tackled by our INDESIT crawler.

The problem of retrieving documents that are "relevant" to a user's information need is the main objective of the information retrieval field [18]. Although our problem is different in nature, in our method we exploit state-of-the-art keyword extraction and term weighting results from IR [18].

There are several recent research projects that address issues related to ours. The goal of CIMPLE is to develop a platform to support the information needs of the members of a virtual community [13]. Compared to our method, CIMPLE requires an expert to provide a set of relevant sources and to design an entity relationship model describing the domain of interest. The MetaQuerier developed by Chang *et al.* has similar objectives to our proposal, as it aims at supporting exploration and integration of databases on the web [10]. However it concentrates on the deep-web.

A new data integration architecture for web data is the subject of the PAYGO project [17]; the project focuses on the heterogeneity of structured data on the web: it concentrates on explicit structured sources, such as Google Base and the schema annotations of Google Co-op, while our approach aims at finding data rich pages containing information of interest. Somehow, our approach can be seen as a service for populating the data sources over which PAYGO works.

Cafarella et al. are developing a system to populate a probabilistic database with data extracted from the web [8]. Data extraction is performed by TEXTRUNNER [3], an information extraction system which is not suitable for working on data rich web pages that are the target of our searches.

Other related projects are TAP and SEMTAG by Guha *et al.* [14, 12]. TAP involves knowledge extracted from structured web pages and encoded as entities, attributes, and relations. SEMTAG provides a semantic search capability driven by the TAP knowledge base. Contrarily to our approach, TAP requires hand-crafted rules for each site that it crawls, and when the formats of those sites change, the rules need to be updated.

4. SEARCHING PAGES BY STRUCTURE: INDESIT

Given a seed page p_0 containing data of interest, the goal of the INDESIT algorithm [5] is to pick out from its site the largest number of pages similar in structure to p_0 and the anchors pointing to such pages. The underlying idea of INDESIT is that while crawling, it is possible to acquire knowledge about the navigational paths the site provides and to give higher priority to the most promising and efficient paths, i.e. those leading to a large number of pages structurally similar to the seed.

INDESIT relies on a simple model that abstracts the structure of a web page. The model adopted by INDESIT to abstract the structure of a web page is based on the following observations: *(i)* pages from large web sites usually contain a large number of links, and *(ii)* the set of layout and presentation properties associated with the links of a page can provide hints about the structure of the page itself. Therefore, whenever a large majority of the links of two pages share the same layout and presentation properties, then it is

likely that the two pages share the same structure. Based on these observations, in INDESIT the structure of a web page is described by means of the presentation and layout properties of the links that it offers, and the structural similarity between pages is measured with respect to these features.

The page model is used by a crawling algorithm that explores a given Web site toward pages sharing the same structure of an input seed page. The crawler navigates the Web site searching for pages that contain lists of links leading to pages that are structurally similar to the seed page. Since these lists of links work like indexes to the searched pages, the crawler rely on them to reach the target set of pages.

The experimental results of our evaluation are reported in Figure 2 and summarize the experiments in [5]. We report the average recall (R), the average precision (P), and the average number of downloaded pages (#dwnl) over 37 INDESIT executions.

R	P	**#dwnl**
95.31%	96.56%	3,389.22

Figure 2: INDESIT experimental results.

5. SEARCHING ENTITIES ON THE WEB: OUTDESIT

INDESIT searches for entity pages within the same site of the input samples. We now describe how the search of entity pages can be extended on the web. The overall idea is to use the results obtained by a first run of INDESIT on the sample pages in order to issue a number of queries against a search engine, such as Google or Yahoo!, with the objective of finding new sources offering other instances of the same entity. This task is performed by the OUTDESIT algorithm, which is described in Figure 3.

As we are interested in finding instances of the target entity, we need to search the web by means of keywords that works as instance identifiers. Our approach is to extract these identifiers from the results of the previous INDESIT executions. Namely, we use the anchors of links pointing to the pages collected by INDESIT as keywords (lines 10–11 in Figure 3). The rationale is that as web pages are produced for human consumption, the anchors of links pointing to entity pages are likely to be values that univocally identify the target instance. E.g., in our basketball players scenario, the anchor of the links to each player page is the name of the player. Observe that, for the sake of usability, this feature has a general validity on the web. For example, the anchor to a book page usually is the title of the book; the anchor to a stock quote is its name (or a representative symbol), etc..

We leverage this property to run searches on the web (lines 9–22). OUTDESIT launches one search for each new anchor found in the previous INDESIT execution. To better focus the search engine, each query is composed by an anchor plus a set of keywords, that we call the entity description. Observe (line 14) that the query is composed by a conjunction of three terms: *(i)* an anchor; *(ii)* a domain keyword k_E, which characterizes the overall conceptual domain; *(iii)* a disjunction of the keywords terms $t_1, \ldots, t_n$, which describe the conceptual entity. All these keywords are extracted automatically from the sample pages, as described in the following of this section.

Each search produces a number of result pages,[1] which are analyzed with the $isInstance$ function to check whether they represent instances of the target entity (line 17). For each page that is classified as an entity page, a new instance of INDESIT is run (line 18), and the whole process is iterated until new pages are found.

```
Algorithm OUTDESIT
Parameter: N number of iterations
Input: a set of sample pages S = {p_0, ..., p_k}
       containing data about instances of the same conceptual entity
Output: a set of pages about the input conceptual entity;
1. begin
     Let R be a set of result pages;
     Let R = INDESIT(p);
     // apply INDESIT to all input pages in S and
5.   // insert the resulting pages into R
     Let σ_E = {t_1, ..., t_n} be the entity intensional description
         computed from R;
     Let k_E be the domain entity description computed from R;
     for (i=0; i<N, 1++) do begin
10.    Let A be the set of new anchors leading to
           the pages returned by the last INDESIT invocations;
       for all terms a ∈ A do begin
         Let W be the set of pages returned by a search
             engine when looking for a ∧ (t_1 ∨ ... ∨ t_n) ∧ k_E;
15.      for all pages p ∈ W do begin
           if the domain of p has been already visited continue
           if (isInstance(p, σ_E)) begin
             add INDESIT(p) to R
           end
20.      end
       end
     end
   end

Function isInstance
Parameter: t template similarity threshold
Input: a page p,
       an intensional description σ_E of the conceptual entity
Output: true iff p is a page about the searched conceptual entity
begin
   Let I = INDESIT(p);
   if |I| = 1
     return false
   Let T be the set of tokens in the template of I;
   Let D be the set of English terms in T;
   return true iff |σ_E ∩ D| / |σ_E| > t;
end
```

Figure 3: The OUTDESIT algorithm.

A fundamental issue in each iteration is to check whether a page returned by the search engine can be considered as an instance of the target conceptual entity. The search engine can in fact return pages that, though containing the required keywords, are not suitable for our purposes. Typical examples are pages from forums, blog, or news where the keywords occurs by chance, or because they are in a free text description. To control this aspect OUTDESIT requires that the keywords of the entity description appear in the template of the retrieved page.

Then, for each page returned by the search engine, an instance of INDESIT is run to obtain a set of structurally similar pages,[2] and their template is computed. If the computed template contains the keywords of the entity description, the page is considered valid; otherwise it is discarded.

Valid pages are finally used as seeds for new INDESIT scans, thus contributing to further discover new pages in the iterative step performed by OUTDESIT.

[1]For each search, we take the first 30 result pages returned by the search engine.

[2]In this step, we run a "light" version of INDESIT, which quickly returns a small set of pages.

5.1 Learning the Entity Description

The description of an entity E, is composed by an *intensional description* and by a *domain keyword*. The intensional description, denoted σ_E, consists of a set of terms $\sigma_E = \{t_1, t_2, \ldots, t_n\}$ and is extracted from the sample pages by analyzing the terms that occur in their templates. The domain keyword, denoted k_E, characterizes general features of the entity and is generated by adapting in our context standard keyword extraction techniques.

Extraction of the Intensional Description. Our approach for generating the set of keywords to be associated with the conceptual entity is based on the observation that pages from large web sites are built over a template that usually contains labels describing the semantics of the data presented in the pages. Consider again the three basketball player pages in Figure 1 and observe labels such as *weight*, *height*, *position*, *college*: they are used by the page designers to provide a meaning to the published data.

Our method for extracting a characterizing description of the entity is based on the assumption that instances of the same conceptual entity have data that refer to a core set of common attributes, even these from different sources. For example, it is likely that most of the instances of the BASKETBALLPLAYER conceptual entity present fields to describe *height*, *weight* and *college* data. This is a strong yet realistic assumption; in their studies on web scale data integration issues, Madhavan *et al.* observe that in the huge repository of Google Base, a recent offering from Google that allows users to upload structured data into Google, *"there is a core set of attributes that appear in a large number of items"* [17].[3] Also, in web pages, these data are usually accompanied by explicative labels, and then they belong to the page template. For example, in the three sample pages shown in Figure 1 (it is worth saying that these pages have been randomly chosen from the web) there are several labels that are present in all three pages. Our method aims at catching these labels to characterize the description of the target entity. To this end, we first compute terms that do belong to the page templates of the sample pages. Then, we choose, as characterizing keywords, those that appear in all the templates.

To illustrate our solution for extracting terms from the page template it is convenient to consider a web page as a sequence of tokens, where each token is either a HTML tag or a term (typically an English word). Each token t is associated a *path*, denoted $path(t)$, which corresponds to the associated path in the DOM tree. Two tokens are equal if they have the same path. In the following, for the sake of readability, we may blur the distinction between token and path associated with the token, assuming that different tokens have different paths.

To detect tokens from the template of a given page we have adapted in our context a technique proposed by Arasu and Garcia-Molina [2]. They observe that given a set of pages P generated by the same template, sets of tokens having the same path and frequency of occurrence in every page in P are likely to belong to the page template.

Let us introduce an example to show how we use these sets to infer a conceptual entity description. Figure 4 shows the sequence of tokens corresponding to three pages in Figure 1. The set of tokens whose paths occur exactly once is given by: `Weight`, `Profile`, `<TR>`, `<TABLE>`, `<B>`. It is reasonable to assume that they belongs to the template that originated the three pages.

The above condition allows us to discover template elements, but it might not hold if a token belonging to the template coincides (by chance) with some other token appearing in some page; for example with an instantiated value embedded in the template. However, observe that if the tokens that occur once in all the pages can be considered template's elements, it is reasonable that they indicate delimiters of homogeneous page segments, i.e. segments generated by the same piece of the underlying template. Then it is possible to inspect each segment, in order to further discover new template tokens. Occurrences of tokens that are not unique on the original set of pages could become unique within the more focused context of a segment. To illustrate this point, let us continue with the previous example: observe that the token `Height`, which is likely to belong to the page template, cannot be included in the computed set, because it occurs twice in the second page (it appears in the profile of the player described in that page). But consider the segments of pages delimited by the tokens detected in the previous step: the token `Height` occurs once in the second segment of every page, which delimited by the tokens `Weight` and `<TABLE>`.

Algorithm TEMPLATETOKENS
Input: a set of token sequences $\mathcal{S} = \{s_1, \ldots, s_n\}$
Output: a set of tokens
begin
  Let $\mathcal{T}$ be an empty set of tokens;
  Let $\mathcal{E}_0 = \{e_1, \ldots, e_k\}$ be the list of tokens
    that occur exactly once in every element of $\mathcal{S}$;
  for each token $e_i \in \mathcal{E}_0$ **do begin**
    Let $S^i = \{s_1^i, \ldots, s_n^i\}$ be a set of sequences such
      that $s_j^i = subSequence(s_j, \mathcal{E}, e_i)\ \forall j = 1, \ldots, n$;
    add $TemplateTokens(S^i)$ to $\mathcal{T}$;
  end
  return $\mathcal{T}$;
end

Function $subSequence(s, \mathcal{E}, e_i)$
Input: s a sequence of tokens $s = t_0 \cdot \ldots \cdot t_n$
  $\mathcal{E}$ a list of tokens $e_0, \ldots, e_k, e_i \in s\ \forall i = 1, \ldots, k$
  e_i a token, $e \in \mathcal{E}$
Output: a subsequence of s
begin
  Let i be the index of e_i in s;
  if (i==0) **begin**
    $start = 0$;
    $end = index - 1$;
  end
  if (i==k) **begin**
    $start = i + 1$;
    $end = n$;
  end
  else begin
    $start = i + 1$;
    Let end be the index of e_{i+1} in s;
    $end = end - 1$;
  end
  return $t_{start} \cdot \ldots \cdot t_{end}$;
end

Figure 5: The TEMPLATETOKENS algorithm to detect tokens belonging to the template of a set of pages.

Given a set of pages, the set of tokens that are likely to belong to the template are computed using the TEMPLATETOKENS algorithm in Figure 5. The algorithm extracts tokens occurring once and uses them to segment the input pages. Segments are then recursively processed to discover other template tokens. The English terms contained in the set of tokens returned by TEMPLATETOKENS are

[3] In the Google Base terminology, an *item* corresponds to a set of attribute-value pairs.

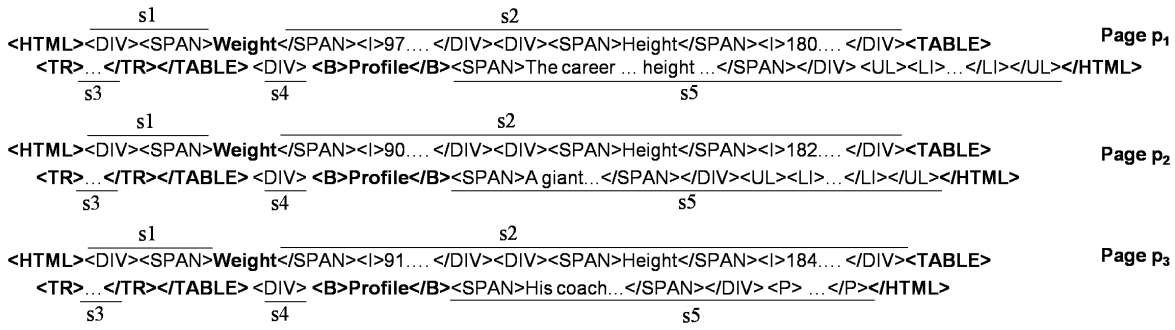

Figure 4: Pages as sequences of tokens.

likely to belong to the template of the input page. However some of them could be originated also by that portion of the template that is usually shared by every page in a site (comprehending page portions such as headers, footers, navigational bars, and so on). To eliminate these terms, we apply the TEMPLATETOKENS algorithm over a broader set of pages, which includes the home page of the sample page site. The terms returned by this execution are then subtracted from the set of terms found in the template of the instance pages. This procedure is performed for each sample page. Finally, in order to obtain the core of terms that is shared by instance pages from different sources, we compute the intersection among the sets of terms computed from each sample.[4] We report in Figure 6 some examples of the entity description generated using our tool.

DOMAIN	*attributes*
BASKETBALL	*pts, height, weight, min, ast*
GOLF	*college, events, height, season, weight*
HOCKEY	*born, height, log, round, shoots, weight*
SOCCER	*club, height, nationality, weight*

Figure 6: Generated descriptions for four conceptual entities.

Domain Keyword Extraction. Our approach for extracting a keyword characterizing the conceptual domain of the entity represented by the sample pages is rather standard. We compute the intersection among the terms that appear in all the sample pages and in the home pages of their sites. The goal is to extract the keywords that most frequently occur in the web sites of the samples. The resulting set of terms are then weighted with the standard TF-IDF scheme [18]. In particular, we consider the term frequency of each term t as the occurrences of the term in the whole set of pages including the samples and the home pages of their sites. To compute the IDF factor, we consider the estimated occurrence of the t on the web, as reported in the web Term Document Frequency and Rank service of the UC Berkeley Digital Library Project. The term with the highest weight is then associated to the entity description. In our example, the term "basketball" is associated to the BASKETBALLPLAYER conceptual entity.

6. EXPERIMENTS

We have developed a prototype that implements OUTDESIT and we have used it to perform some experiments to validate our techniques.

[4]The resulting set is also polished by removing terms that do not correspond to English nouns.

We have focused our experiments on the sport domain. The motivation of our choice is that it is easy to interpret the published information, and then to evaluate the precision of the results produced by our method. The goal of our experiments was to search for a set of pages, each one containing data about one athlete (player) of a given sportive discipline. We have concentrated on four disciplines: basketball, soccer, hockey, and golf. Therefore, we may say that our experiments aimed at discovering pages publishing data about instances of the following conceptual entities: BASKETBALLPLAYER, SOCCERPLAYER, HOCKEYPLAYER, and GOLFPLAYER.

For each discipline we have taken three sample pages, from three different web sites, each one publishing data about one player of that discipline. Then, for each sample set we have run OUTDESIT. In the following we presents the results of this activity.

6.1 Conceptual Entity Description

Extracted Intensional Descriptions. The results of the entity descriptions generation are reported in Figure 6. A first observation is that all the terms may actually represent reasonable attribute names for the corresponding player entity. Also, we notice that there is a core set of terms which is shared by athletes from different disciplines (namely, *height* and *weight*). Since our experiments involve a taxonomy of the athlete category, it is reasonable that athletes of different sports are described by a core set of attributes.

Extracted Domain Keywords. Figure 7 presents the keywords extracted from each set of sample pages.[5] Observe that the keywords with the greatest weight correctly characterize the domain (they actually correspond to the sport discipline). The domain keyword plays a fundamental role in the OUTDESIT iterations. First, as it is used to generate a more constrained query for the search engine, it allows the system to elaborate a smaller (and more pertinent) set of pages. Second, in case of homonymous athletes involved in different disciplines, the presence of the domain keyword in the query can constrain the search towards the right discipline.

Using Entity Descriptions. We have manually analyzed the behavior of the $isInstance()$ function, which uses the entity description to check whether a given page is valid for our purposes. We have run a single iteration of OUTDESIT with a set of anchors

[5]We only show terms for which the TF-IDF weight is at least 30% of the maximum.

DOMAIN			
keyword	TF	IDF	TF-IDF
BASKETBALL			
basketball	29.0	5.61	162.89
season	27.0	5.08	137.39
team	24.0	4.07	97.86
players	14.0	5.30	74.26
GOLF			
golf	64.0	5.29	338.63
leaderboard	17.0	10.29	175.07
stats	26.0	5.65	147.06
players	25.0	5.30	132.62
HOCKEY			
hockey	22.0	6.30	138.68
teams	11.0	5.26	57.90
SOCCER			
soccer	28.0	5.59	156.62

Figure 7: Extracted keywords.

pointing to 500 SOCCERPLAYER pages, selected randomly from 10 soccer web sites. The search engine returned about 15000 pages distributed over about 4000 distinct web sites. We have then manually evaluated the web sites to measure the precision and the recall of the $isInstance()$ function over the pages returned by the search engine. In particular, we studied how precision and recall behave varying the value for the threshold t in the OUTDESIT algorithm.

As expected, we can see In Figure 8 how raising the threshold the precision increases and the recall decreases. The system achieves a 100% precision when the number of keywords from the description required to be in the template of the page under evaluation is at least 75%. When only 50% of the keywords are required, the pages marked as valid are 74% of the total valid pages returned by the search engine, and the precision is still high at 72%.

It is interesting to notice that only 20% of the web pages returned by the search engine were pages whose data describe instances of the same conceptual entity exemplified by the sample pages. An example of non valid pages that frequently occurred in results returned by the search engine are personal pages (blog), news or forum pages: they are pertinent with the keywords passed to the search engine, but they are not instances of the conceptual entity as in our definition. It is worth saying that some of these pages also contained terms of the intensional description. However, these terms did not appear in the page template as required by our function, and then these pages were correctly discarded.

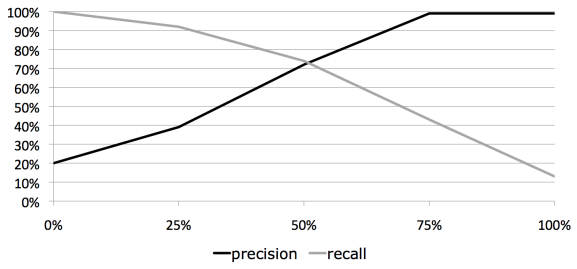

Figure 8: Performance of the $isInstance()$ function varying the threshold t.

6.2 Quantitative Evaluation

The number of pages discovered by OUTDESIT for our four target entities are depicted in Figure 9. Each graph plots the number of new instance pages against the number of new web sites discovered by OUTDESIT. In order to have comparable results, we have run two iterations for each discipline.

Starting from three sample pages, for each conceptual entity our method automatically discovered several thousands of pages. By a manual inspection, conducted on a representative subset of the results, we can conclude that all the retrieved pages can be considered as instances of the entity exemplified by the input sample pages.

The graphs also plot the number of distinct anchors that are found in each step. Somehow they can approximate the number of distinct players. As expected, it is evident that they increase less than the number of pages.

7. CONCLUSIONS AND FUTURE WORK

We have presented a method to automatically discover pages publishing data about a certain conceptual entity, given as input only a small set of sample pages.

The set of pages retrieved by our method can be used to build a custom, entity aware search engine. As a proof of concept, we are building entity aware search engines for sport fans.[6] To this end we are populating a Google Co-op search engine with the pages retrieved by OUTDESIT in our experiments. Each page is associated with an annotation (*facet* in the Google Co-op terminology) corresponding to the name of the entity exploited by OUTDESIT. Users can use these annotations to semantically refine the query results by restricting the search towards pages associated with the annotation.

The results of the experimental activity suggest improvements that need to be developed, as well as new intriguing research directions. The current method for learning the entity description is a bit simplistic. In particular, we have observed that computing the keywords in the bootstrapping phase is too rigid. We are therefore developing a novel method, based on a probabilistic model, to dynamically compute a weight for the terms of the templates of the pages that OUTDESIT retrieves. Another issue that need to be addressed to improve our approach deals with the development of methods to consider also pages from the hidden web: techniques such as those proposed for building focused crawlers for the hidden web could be profitably adapted in our context [15, 4].

An important research direction that we are investigating is the extension of automatic wrapping techniques (such as those proposed in [11] and [2]) to extract, mine and integrate data from the sources OUTDESIT retrieves, as we have recently demonstrated [6]. Another interesting study deals with the development of record linkage techniques for the instances retrieved by our system. We believe that our method, which progressively discovers new instances from previously achieved results, can provide an interesting basis for new approaches. Finally, we believe that a challenging issue is to study extensions of our framework in order to take into account also relationships among different entities.

8. REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *DL*,2000.

[2] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD*, 2003.

[3] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[4] L. Barbosa, and J. Freire. An adaptive crawler for locating hiddenwebentry points. *WWW*, 2007.

[6] http://flint.dia.uniroma3.it/

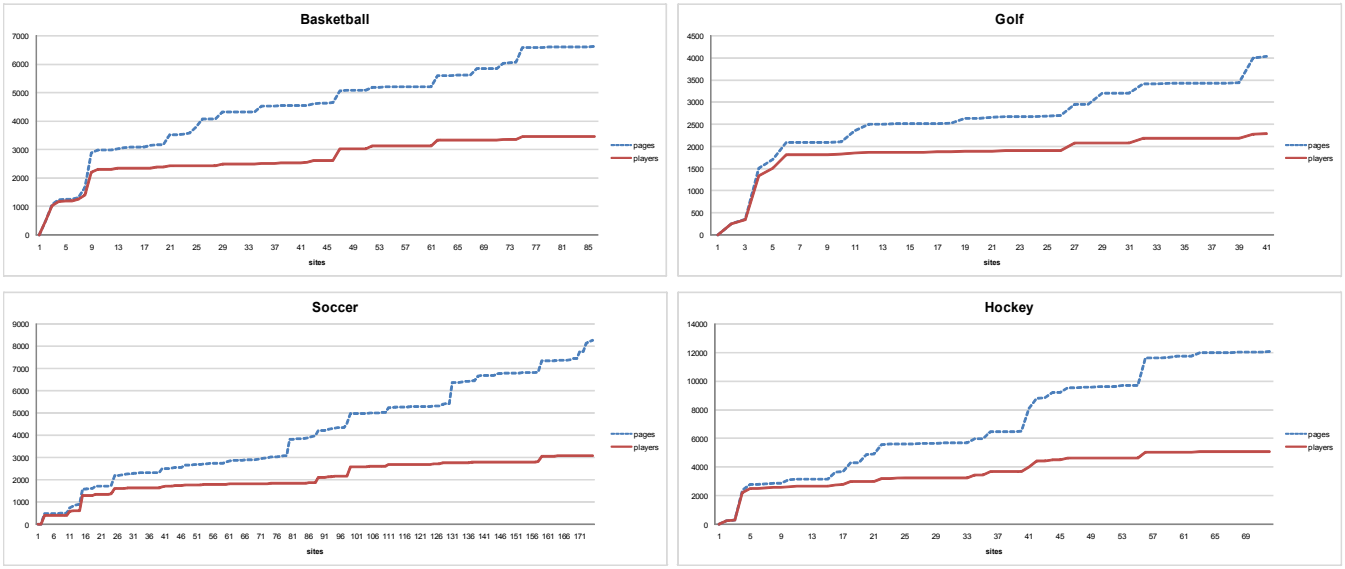

Figure 9: Pages and players found by OUTDESIT.

[5] L. Blanco, V. Crescenzi, and P. Merialdo. Efficiently locating collections of web pages to wrap. In *WEBIST*, 2005.

[6] L. Blanco, V. Crescenzi, P. Merialdo, and P. Papotti. Flint: Google-basing the Web. In *EDBT (Demo section)*, 2008.

[7] S. Brin. Extracting patterns and relations from the World Wide Web. In *WebDB*, 1998.

[8] M. J. Cafarella, O. Etzioni, and D. Suciu. Structured queries over web text. *IEEE Data Eng. Bull.*, 29(4):45–51, 2006.

[9] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

[10] K. C.-C. Chang, B. He, and Z. Zhen. Toward large scale integration: Building a metaquerier over databases on the web. In *CIDR*, 2005.

[11] V. Crescenzi, G. Mecca, and P. Merialdo. ROADRUNNER: Towards automatic data extraction from large Web sites. In *VLDB*, 2001.

[12] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *WWW*, 2003.

[13] A. Doan, R. Ramakrishnan, F. Chen, P. DeRose, Y. Lee, R. McCann, M. Sayyadian, and W. Shen. Community information management. *IEEE Data Eng. Bull.*, 29(1):64–72, 2006.

[14] R. Guha and R. McCool. Tap: a semantic web platform. *Computer Networks*, 42(5):557–577, August 2003.

[15] B. He, and K. C.-C. Chang. Statistical Schema Matching across Web Query Interfaces. *SIGMOD* 2003.

[16] D. Hand, H. Mannila, and S. P. *Principles of Data Mining*. MIT Press, 2001.

[17] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, 2007.

[18] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[19] G. Pant and P. Srinivasan. *Learning to crawl: Comparing classification schemes*. *ACM Trans. Inf. Syst.*, 23(4):430–462, 2005.

[20] S. Sizov, M. Theobald, S. Siersdorfer, G. Weikum, J. Graupmann, M. Biwer, and P. Zimmer. The bingo! system for information portal generation and expert web search. In *CIDR*, 2003.

[21] C. Van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.

[22] M. L. A. Vidal, A. Soares da Silva, E. Silva de Moura, and J. M. B. Cavalcanti. Structure-driven crawler generation by example. *SIGIR*, 2006.